September 1998

Equality, Type and Word Constraints

Peter Buneman
University of Pennsylvania

Wenfei Fan
University of Pennsylvania

Scott Weinstein
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-98-32.

Equality, Type and Word Constraints

Abstract

As a generalization of inclusion dependencies that are found in relational databases, word constraints have been studied for semistructured data [6] as well as for an object-oriented model [10]. In both contexts, it is assumed that each data entity has a unique identity, and two entities are equal if and only if they have the same identify. In this setting the decidability of the implication and finite implication problems for word constraints has been established. A question left open is whether these problems are still decidable in the context of an object-oriented model $M^{\approx}$ which supports complex values with nested structures and complex value equality. This paper provides an answer to that question. We characterize a schema in $M^{\approx}$ in terms of a type constraint and an equality constraint, and investigate the interaction between these constraints and word constraints. We show that in the presence of equality and type constraint, the implication and finite implication problems for word constraints are also decidable by giving a small model argument.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-98-32.

Equality, Type and Word Constraints

Peter Buneman[*] peter@central.cis.upenn.edu
Wenfei Fan[†] wfan@saul.cis.upenn.edu
Scott Weinstein[‡] weinstein@linc.cis.upenn.edu

Department of Computer and Information Science
University of Pennsylvania

September 1998

Abstract

As a generalization of inclusion dependencies that are found in relational databases, word constraints have been studied for semistructured data [6] as well as for an object-oriented model [10]. In both contexts, it is assumed that each data entity has a unique identity, and two entities are equal if and only if they have the same identify. In this setting, the decidability of the implication and finite implication problems for word constraints has been established. A question left open is whether these problems are still decidable in the context of an object-oriented model $\mathcal{M}^{\approx}$ which supports complex values with nested structures and complex value equality. This paper provides an answer to that question. We characterize a schema in $\mathcal{M}^{\approx}$ in terms of a type constraint and an equality constraint, and investigate the interaction between these constraints and word constraints. We show that in the presence of equality and type constraint, the implication and finite implication problems for word constraints are also decidable, by giving a small model argument.

1 Introduction

Word constraints were introduced in [6] to generalize inclusion dependencies that are commonly found in relational databases. They are useful for, among others, query optimization in a variety of database contexts, ranging from semistructured data to object-oriented databases. In these contexts, a database is modeled as a rooted edge-labeled directed graph, and a word constraint is defined to be a first-order logic sentence of the form:

$$\forall\, x\, (\alpha(r, x) \rightarrow \beta(r, x))$$

[*]Supported partly by the Army Research Office (DAAH04-95-1-0169) and NSF Grant CCR92-16122.
[†]Supported by a graduate fellowship from the Institute For Research in Cognitive Science, University of Pennsylvania.
[‡]Supported by NSF Grant CCR-9403447.

Here r is a constant denoting the root of the graph, and $\alpha(r, x)$ and $\beta(r, x)$ are paths from r to x, which can be represented as logic formulas by taking edge labels as binary predicates. For example, the following are word constraints for a (semistructured) school database, taken from [10]:

$$\begin{aligned} \forall\, x\, (students \cdot taking(r, x) &\rightarrow courses(r, x)) \\ \forall\, x\, (courses \cdot taken\_by(r, x) &\rightarrow students(r, x)) \end{aligned}$$

These constraints state that every course which is being taken by a student in the database must be a course in the database, and similarly, every student who is taking a course in the database must be a student in the database. In first-order logic, these constraints can be expressed as:

$$\begin{aligned} \forall\, x\, (\exists\, y\, (students(r, y) \wedge taking(y, x)) &\rightarrow courses(r, x)) \\ \forall\, x\, (\exists\, y\, (courses(r, y) \wedge taken\_by(y, x)) &\rightarrow students(r, x)) \end{aligned}$$

The implication and finite implication problems, which are the central questions for word constraints, have been studied for semistructured data [6] and an object-oriented model $\mathcal{M}$ [9, 10]. Semistructured data, e.g., as found in the Web, is modeled as a rooted edge-labeled directed graph, unconstrained by any type system or schema [1, 8]. As in OEM [5], it is assumed that each data entity (node) in the graph has a unique identity, and two entities are equal if and only if they have the same identity. This equality relation is called *identity equality*. The model $\mathcal{M}$ is similar to the one studied in [2]. It supports atomic types, classes, and the record and finite set constructs. A database of $\mathcal{M}$ can be viewed as a collection of object-identifiers (oids). The value of an oid is either a basic value (i.e., a value of an atomic type), a record consisting of basic values and oids, or a set consisting of basic values or oids. In other words, $\mathcal{M}$ does not support complex values with nested structures such as sets of records or records with set component. In $\mathcal{M}$, the equality relation is also defined to be identity equality. A schema in $\mathcal{M}$ can be viewed as imposing a type constraint on the data, and a database instance of the schema can be understood as a semistructured database satisfying the type constraint. In both $\mathcal{M}$ and the semistructured data model, the decidability of word constraint implication has been established in [6, 9, 10].

In many object-oriented database systems such as those studied in [3, 4, 12, 15], complex values with nested structures are common. Such a complex value may not have a unique identity. Consequently, the equality relation on complex values cannot be simply treated as identity equality. More specifically, the equality relation in these systems is called *complex value equality*, denoted by $\approx$ and defined as follows:

1. Let v_1 and v_2 be values of an atomic type. Then $v_1 \approx v_2$ if $v_1 = v_2$ (i.e., v_1 and v_2 are identical).

2. Let o_1 and o_2 be objects of a class. Then $o_1 \approx o_2$ if o_1 and o_2 have the same oid. That is, equality on objects is defined by comparing object identities.

3. Let s_1 and s_2 be of a set type. Then $s_1 \approx s_2$ if for every $x \in s_1$, there is $y \in s_2$ such that $x \approx y$, and vice versa.

4. Let r_1 and r_2 be values of type $record(l_1 : \tau_1, \ldots, l_n : \tau_n)$. Then $r_1 \approx r_2$ if for every $i \in [1, n]$, $r_1.l_i \approx r_2.l_i$. Here $v.l$ stands for the projection of v at attribute l.

A natural question to answer here is whether word constraint implication is still decidable in the context of a simple extension of $\mathcal{M}$, $\mathcal{M}^{\approx}$, which supports complex values with nested structures. In other words, when complex value equality is supported, whether is word constraint implication still decidable? One may question the need to re-investigate the problem since at first glance, complex value equality appears to have no severe impact on word constraint implication. This appearance can be dispelled by considering the following example.

Example 1.1: Let Δ be a schema in $\mathcal{M}$ and Δ' a schema in $\mathcal{M}^{\approx}$:

```
Δ:  class player {
          type record (Name: string,
                       Rank: int)
    }

    class team {
          type set(player)
    }

    Team1, Team2:  team;


Δ′:  class player    /* as defined above

     Team1, Team2:  set(player);
```

A database instance of Δ is a value of database type ($DBtype$):

$$record(Team1 : team,\ Team2 : team).$$

Similarly, a database instance of Δ' is a value of $DBtype$:

$$record(Team1 : set(player),\ Team2 : set(player)),$$

which is a complex value with nested structure. Because of this, Δ' is not a schema in $\mathcal{M}$.

Let `Jones` and `Smith` be objects of `player`, created by:

```
Jones = new player("Jones", 2);
Smith = new player("Smith", 3);
```

Using these objects, an instance I of Δ and an instance I' of Δ' are given as follows:

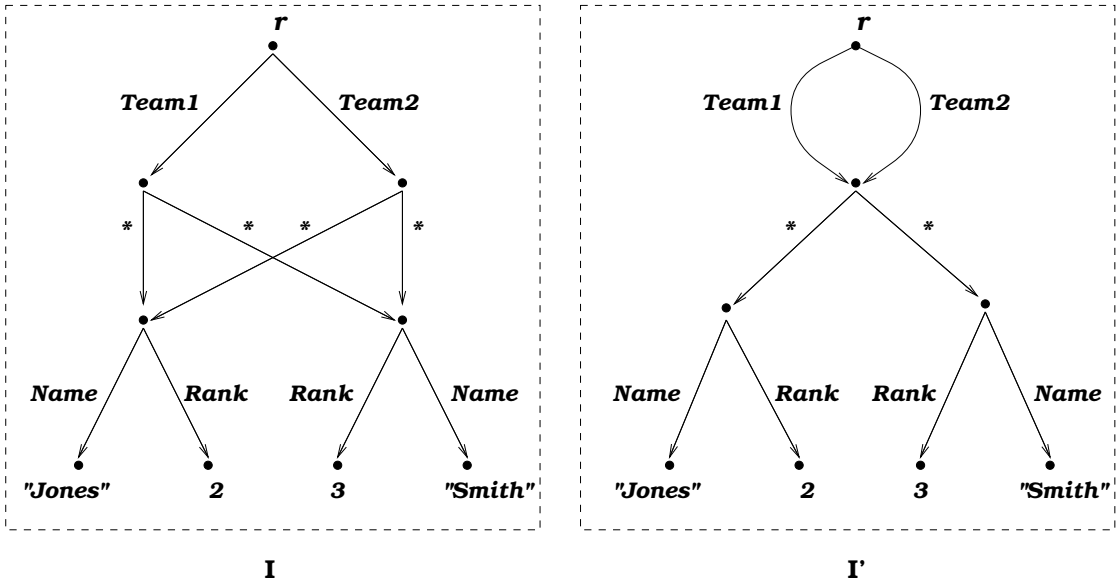

Figure 1: Example databases

I: `Team1 = new team(Jones, Smith);`
`Team2 = new team(Jones, Smith);`

I': `Team1 = set(Jones, Smith);`
`Team2 = set(Jones, Smith);`

The databases I and I' are represented by the graphs shown in Figure 1, in which r denotes the root and $*$ denotes set membership.

Now let us consider the following word constraints:

$$\begin{aligned}
\varphi_1 &= \forall\, x\,(\exists\, y\,(Team1(r,y) \wedge *(y,x)) \rightarrow \exists\, y\,(Team2(r,y) \wedge *(y,x))) \\
\varphi_2 &= \forall\, x\,(\exists\, y\,(Team2(r,y) \wedge *(y,x)) \rightarrow \exists\, y\,(Team1(r,y) \wedge *(y,x))) \\
\varphi &= \forall\, x\,(Team1(r,x) \rightarrow Team2(r,x))
\end{aligned}$$

Here φ_1 and φ_2 ensure that `Team1` and `Team2` consist of the same players. Constraint φ in fact states that `Team1` and `Team2` are equal, since $DBtype$ is a record type in both Δ and Δ'. It is easy to see that $I \models \varphi_1 \wedge \varphi_2$ but $I \not\models \varphi$. This is because in I, `Team1` and `Team2` are objects with different oids even if they have the same value. However, $I' \models \varphi_1 \wedge \varphi_2$ and $I' \models \varphi$. The reason is that in I', `Team1` and `Team2` are defined to be sets. As a result, they are equal as long as they consist of the same players. In fact, it can be shown that in the context of database instances of Δ', $\{\varphi_1, \varphi_2\}$ indeed implies φ. In contrast, in the context of databases of any schema in $\mathcal{M}$, if φ_1, φ_2 and φ are defined, then $\{\varphi_1, \varphi_2\}$ does not imply φ.

This example shows that the difference between the definitions of equality in $\mathcal{M}$ and $\mathcal{M}^{\approx}$

gives rise to different outcomes of word constraint implication. ∎

As illustrated by the example above, complex value equality interacts with word constraints. In other words, it does make life harder. In the context of $\mathcal{M}$, it has been shown that a schema can be characterized in terms of a type constraint [9, 10]. The interaction between type constraints and path constraints, which is a class of constraint more general than word constraints, has been well studied [11]. However, in the presence of complex value equality, an equality constraint is also needed, in addition to type constraint, to capture the semantics of a schema. The interaction of the three different forms of constraints – equality, type and word constraints – has not been addressed.

This paper investigates the interaction of equality, type and word constraints. To focus on the central issue and to simplify the discussion, we define the object-oriented model $\mathcal{M}^{\approx}$ in the flavor of the nested relational model [3]. That is, set and record constructors are required to alternate (i.e., set of sets and record with a record component are prohibited). We characterize a schema Δ in $\mathcal{M}^{\approx}$ in terms of an equality constraint, in addition to the type constraint developed for $\mathcal{M}$ [9, 10]. We represent database instances of Δ as (finite) logic structures satisfying both the type constraint and the equality constraint. Using this abstraction, we show that the implication and finite implication problems for word constraints are decidable in the context of $\mathcal{M}^{\approx}$. More specifically, we present an elementary proof of the decidability by giving a small model argument. That is, given a finite set $\Sigma \cup \{\varphi\}$ of word constraints, we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model, then it has a model of size at most exponential in the length of $\bigwedge \Sigma \wedge \neg\varphi$.

It should be noted that complex value equality is a notion different from the so called shallow and deep equalities [2]. Shallow and deep equalities on objects are not oid equality. In particular, the definition of deep equality is recursive and thus is not definable in first-order logic [2]. To clarify the difference, below we give the definitions of these predicates.

- *Shallow equality*, denoted $\sim_s$, is defined in the same way as value equality except for the case of objects, which is given as follows. Let o_1 and o_2 be objects of a class, and v_1, v_2 be the values of o_1 and o_2, respectively. Then $o_1 \sim_s o_2$ if $v_1 \approx v_2$.
- *Deep equality*, denoted $\sim_d$, can be described as follows.
  1. Let v_1 and v_2 be values of an atomic type. Then $v_1 \sim_d v_2$ if $v_1 = v_2$.
  2. Let o_1 and o_2 be objects of a class, and v_1, v_2 be the values of o_1 and o_2, respectively. Then $o_1 \sim_d o_2$ if $v_1 \sim_d v_2$.
  3. Let s_1 and s_2 be of a set type. Then $s_1 \sim_d s_2$ if for every $x \in s_1$, there is $y \in s_2$ such that $x \sim_d y$, and vice versa.
  4. Let r_1 and r_2 be values of type $record(l_1 : \tau_1, \ldots, l_n : \tau_n)$. Then $r_1 \sim_d r_2$ if for every $i \in [1, n]$, $r_1.l_i \sim_d r_2.l_i$.

The rest of the paper is organized as follows. Section 2 presents the object-oriented model $\mathcal{M}^{\approx}$, specifies the type constraint and equality constraint, and gives an abstraction

of databases of $\mathcal{M}^{\approx}$. Section 3 formally defines word constraints in the context of $\mathcal{M}^{\approx}$, and justifies the abstraction of databases given in Section 2 with respect to word constraint implication. Section 4 establishes the decidability of the implication and finite implication problems for word constraints in the context of $\mathcal{M}^{\approx}$. Finally, Section 5 summarizes the main results of the paper.

2 Equality and type constraints for $\mathcal{M}^{\approx}$

In this section, we present the object-oriented model $\mathcal{M}^{\approx}$, define equality and type constraints, and give an abstraction of the databases of $\mathcal{M}^{\approx}$ in terms of these constraints. Finally, we compare $\mathcal{M}^{\approx}$ with the object-oriented model $\mathcal{M}$ studied in [9, 10].

2.1 The data model $\mathcal{M}^{\approx}$

We begin by describing the object-oriented model $\mathcal{M}^{\approx}$.

Assume a fixed countable set of labels, $\mathcal{L}$, and a fixed finite set of *atomic types*, $\mathcal{B}$.

Definition 2.1: Let $\mathcal{C}$ be some finite set of *classes*. The set of *types over* $\mathcal{C}$, denoted $Types^{\mathcal{C}}$, is defined by the following abstract syntax:

$$\begin{array}{rcl} \tau & ::= & b \mid C \mid Set \mid Record \\ Set & ::= & \{b\} \mid \{C\} \\ Record & ::= & [l_1 : \delta, \ldots, l_n : \delta] \\ \delta & ::= & b \mid Set \end{array}$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. The notations $[l_1 : \tau, \ldots, l_n : \tau]$ and $\{\tau\}$ represent *record type* and *set type*, respectively. We reserve τ to range over $Types^{\mathcal{C}}$. ■

Definition 2.2: A *schema* in $\mathcal{M}^{\approx}$ is a triple $\Delta = (\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ is a finite set of classes,
- ν is a mapping: $\mathcal{C} \rightarrow Types^{\mathcal{C}}$ such that for each $C \in \mathcal{C}$, $\nu(C) \in Record$, and
- $DBtype$ is a record type in $Types^{\mathcal{C}}$ having the form:

$$[l_1 : \{\tau_1\}, \ldots, l_n : \{\tau_n\}]$$

■

Here we assume that every database has a unique (persistent) entry point, and $DBtype$ in a schema specifies the type of the entry point. The entry point is a collection of sets. This definition of database schema is in the same spirit as those found in [3, 4, 15].

The following should be noted about the definitions above.

- Given a schema $\Delta = (\mathcal{C}, \nu, DBtype)$ in $\mathcal{M}^{\approx}$, $DBtype$ can be converted into a "pure type", i.e., a type containing no classes, by continuously substituting $\nu(C)$ for C for every $C \in \mathcal{C}$. In this pure type, the set and record constructors alternate. That is, there is no set of sets or record with a record component. This is in the flavor of the nested relational model [3].
- To simplify the discussion, we require that a set consists of either basic values or oids. That is, in $\mathcal{M}^{\approx}$, complex values with nested structures are limited to be values of *Record*. In addition, a record in $\mathcal{M}^{\approx}$ consists of only basic values and sets. This restriction can be removed without affecting the decidability results of the paper.

Example 2.1: The schema Δ' given in Example 1.1 can be described as $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C} = \{player\}$,
- ν maps $player$ to $[Name : string, Rank : int]$, and
- $DBtype = [Team1 : \{player\}, Team2 : \{player\}]$.

■

Example 2.2: Another example schema of $\mathcal{M}^{\approx}$ is $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C} = \{student, course\}$,
- ν is defined by:

$$\begin{aligned} student &\mapsto [name : string, taking : \{course\}] \\ course &\mapsto [cname : string, taken\_by : \{student\}] \end{aligned}$$

- $DBtype = [students : \{student\}, courses : \{course\}]$.

■

Definition 2.3: A *database instance* of schema $(\mathcal{C}, \nu, DBtype)$ is a triple $I = (\pi, \mu, d)$, where

- π is an *oid assignment* that maps each $C \in \mathcal{C}$ to a finite set of oids, $\pi(C)$, such that for all $C, C' \in \mathcal{C}$, $\pi(C) \cap \pi(C') = \emptyset$ if $C \neq C'$;
- for each $C \in \mathcal{C}$, μ maps each oid in $\pi(C)$ to a value in $[\![\nu(C)]\!]_\pi$, where

$$\begin{aligned} [\![b]\!]_\pi &= D_b, \\ [\![C]\!]_\pi &= \pi(C), \\ [\![\{\tau\}]\!]_\pi &= \{V \mid V \subseteq [\![\tau]\!]_\pi, V \text{ is finite}\}, \\ [\![[l_1 : \tau_1, ..., l_n : \tau_n]]\!]_\pi &= \{[l_1 : v_1, ..., l_n : v_n] \mid v_i \in [\![\tau_i]\!]_\pi, i \in [1, n]\}; \end{aligned}$$

  here D_b denotes the domain of atomic type b;

- d is a value in $[\![DBtype]\!]_\pi$, which represents the (persistent) entry point into the database instance.

We denote the set of all database instances of schema Δ by $\mathcal{I}(\Delta)$. ■

Example 2.3: The instance I' of Δ' given in Example 1.1 can be described as (π, μ, d), where

- $\pi(player) = \{Jones, Smith\}$,
- $\mu : player \rightarrow [\![[Name : string,\ Rank : int]]\!]_\pi$ is defined by

$$\begin{aligned} Jones &\mapsto [Name : \text{“Jones”},\ Rank : 2] \\ Smith &\mapsto [Name : \text{“Smith”},\ Rank : 3] \end{aligned}$$

- d is defined to be $[Team1 : \{Jones, Smith\},\ Team2 : \{Jones, Smith\}]$.

■

2.2 Equality and type constraints

Along the same lines as [9, 10], we present an abstraction of databases of $\mathcal{M}^{\approx}$ in terms of first-order logic. We first characterize every schema Δ in $\mathcal{M}^{\approx}$ by means of two first-order logic sentences, called *the type constraint* and *the equality constraint determined by* Δ, respectively. We then represent databases of Δ as (finite) logic structures satisfying the type and equality constraints. Such a structure can be depicted as an edge-labeled rooted directed graph.

We assume the standard notations used in first-order logic [13].

To define type and equality constraints, we first specify the first-order vocabulary determined by a schema. Two important components of the vocabulary are defined as follows.

Definition 2.4: Given a schema $\Delta = (\mathcal{C}, \nu, DBtype)$, *the set of binary relation symbols* $E(\Delta)$ and *the set of types* $T(\Delta)$ *determined by* Δ are defined to be the smallest sets having the following properties.

1. $DBtype \in T(\Delta)$ and $\mathcal{C} \subseteq T(\Delta)$.
2. For every $\tau \in T(\Delta)$,
   - if $\tau = \{\tau'\}$, then $\tau' \in T(\Delta)$ and $* \in E(\Delta)$;
   - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$), then for $i \in [1, n]$, $\tau_i \in T(\Delta)$ and $l_i \in E(\Delta)$.

■

Obviously, both $E(\Delta)$ and $T(\Delta)$ are finite. Note here we use the distinguished symbol $*$ to denote the set membership.

Using $T(\Delta)$ and $E(\Delta)$, we define the vocabulary as follows.

Definition 2.5: The *signature determined by a schema* Δ *in* $\mathcal{M}^{\approx}$ is a quadruple

$$\sigma(\Delta) = (r,\ E(\Delta),\ R(\Delta),\ Q(\Delta)),$$

where

- r is a constant symbol, denoting the root;
- $E(\Delta)$ is the finite set of binary relation symbols defined above, which denote the edge labels;
- $R(\Delta)$ is the finite set of unary relation symbols defined by $\{R_\tau \mid \tau \in T(\Delta)\}$, which denote the sorts; and
- $Q(\Delta)$ is the finite set of binary relation symbols defined by $\{\approx_\tau \mid \tau \in T(\Delta)\}$, which denote the equality symbols of different sorts.

■

Example 2.4: The signature determined by the schema given in Example 2.2 is $(r,\ E,\ R,\ Q)$, where

- r is a constant, which in each instance $(\pi,\ \mu,\ d)$ of the schema intends to name d;
- $E = \{*,\ students,\ courses,\ name,\ taking,\ cname,\ taken\_by\}$;
- $R = \{R_\tau \mid \tau \in T\}$ and $Q = \{\approx_\tau \mid \tau \in T\}$, where

$$T = \{DBtype,\ student,\ course,\ string,\ \{student\},\ \{course\}\}.$$

■

Given the vocabulary $\sigma(\Delta)$ determined by schema Δ, we specify the type constraint and equality constraint determined by Δ. To do this we use the counting quantifier $\exists!$, whose semantics is described as follows: structure G satisfies $\exists! x\, \psi(x)$ if and only if there exists a unique element a of G such that $G \models \psi(a)$. See [7] for detailed discussions of counting quantifiers.

Definition 2.6: Let Δ be a schema in $\mathcal{M}^{\approx}$. For every τ in $T(\Delta)$, the *type constraint determined by* τ is the logic sentence $\forall\, x\, \phi_\tau(x)$ defined as follows.

- If $\tau = b$, then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)} \neg l(x, y)).$$
- If $\tau = \{\tau'\}$, then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)\setminus\{*\}} \neg l(x, y)) \ \wedge \forall\, y\, (*(x, y) \rightarrow R_{\tau'}(y)).$$

- If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$, then $\phi_\tau(x)$ is

$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{l_1, \ldots, l_n\}} \neg l(x, y)) \ \wedge \bigwedge_{i \in [1,n]} (\exists !\, y\, l_i(x, y) \ \wedge \forall\, y\, (l_i(x, y) \rightarrow R_{\tau_i}(y))).$$

The *type constraint determined by* Δ is the sentence $\Phi(\Delta)$ defined by

$$R_{DBtype}(r) \ \wedge \bigwedge_{\tau \in T(\Delta)} \forall\, x\, \phi_\tau(x) \ \wedge \ \forall\, x\, (\bigvee_{\tau \in T(\Delta)} R_\tau(x) \ \wedge \bigwedge_{\tau \in T(\Delta)} (R_\tau(x) \rightarrow \bigwedge_{\tau' \in T(\Delta) \setminus \{\tau\}} \neg R_{\tau'}(x))).$$

■

Definition 2.7: Let Δ be a schema in $\mathcal{M}^{\approx}$. For every τ in $T(\Delta)$, the *equality constraint determined by* τ is the sentence $\forall\, x\, y\, \phi^{\approx}_\tau(x, y)$ defined as follows.

- If $\tau = b$ or $\tau = C$ for some $C \in \mathcal{C}$, then $\phi^{\approx}_\tau(x, y)$ is

$$x \approx_\tau y \ \leftrightarrow \ x = y.$$

- If $\tau = \{\tau'\}$, then $\phi^{\approx}_\tau(x, y)$ is

$$x \approx_\tau y \ \leftrightarrow \ R_\tau(x) \wedge R_\tau(y) \wedge \ \forall\, z_1\, \exists\, z_2\, (*(x, z_1) \rightarrow *(y, z_2) \wedge z_1 \approx_{\tau'} z_2) \wedge \forall\, z_1\, \exists\, z_2\, (*(y, z_1) \rightarrow *(x, z_2) \wedge z_1 \approx_{\tau'} z_2).$$

- If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\phi^{\approx}_\tau(x, y)$ is

$$x \approx_\tau y \ \leftrightarrow \ R_\tau(x) \wedge R_\tau(y) \wedge \bigwedge_{i \in [1,n]} \forall\, z_1 \forall\, z_2\, (l_i(x, z_1) \wedge l_i(y, z_2) \rightarrow z_1 \approx_{\tau_i} z_2).$$

The *equality constraint determined by* Δ is the sentence $\Phi^{\approx}(\Delta)$ defined by

$$\bigwedge_{\tau \in T(\Delta)} \forall\, x\, y\, \phi^{\approx}_\tau(x, y) \ \wedge \ \forall x\, y\, (\bigvee_{\tau \in T(\Delta)} x \approx_\tau y \ \rightarrow \ x = y).$$

■

The following should be noted.

- For each $\tau \in T(\Delta)$, the definition of $\approx_\tau$ is not recursive, i.e., $\approx_\tau$ is not defined in terms of itself. This is because in $\mathcal{M}^{\approx}$, only classes can be defined recursively, and the equality on objects is defined by comparing oids.
- The type constraint is in two-variable logic with counting, C^2 (see [7] for discussions of C^2). In contrast, the equality constraint cannot be expressed in C^2.
- Both type and equality constraints are definable in first-order logic.

2.3 An abstraction of databases of $\mathcal{M}^{\approx}$

Using the type and equality constraints defined above, we give an abstraction of the databases of $\mathcal{M}^{\approx}$.

Definition 2.8: An *abstract database of a schema* Δ is a finite $\sigma(\Delta)$-structure G such that $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$.

We denote the set of all abstract databases of schema Δ by $\mathcal{U}_f(\Delta)$.

We use $\mathcal{U}(\Delta)$ to denote the set of all the $\sigma(\Delta)$-structures satisfying the following conditions: for each $G \in \mathcal{U}(\Delta)$,

- $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$; and
- G respects *the finite set rule*. That is, for each set type $\tau \in T(\Delta)$ and for each $o \in R_\tau^G$, there are only finitely many o' in G such that $G \models *(o, o')$. As a result, each node in G has finitely many outgoing edges.

■

It should be noted that $\mathcal{U}(\Delta)$ is not definable in first-order logic.

The justification of the abstraction will be given in the next section.

2.4 Comparison of $\mathcal{M}^{\approx}$ with $\mathcal{M}$

Finally, we compare $\mathcal{M}^{\approx}$ with $\mathcal{M}$.

In contrast to $\mathcal{M}^{\approx}$, $\mathcal{M}$ does not support complex values with nested structures. More specifically, assume $\mathcal{L}$, $\mathcal{B}$ and $\mathcal{C}$ as in Section 2.1. The set of types over $\mathcal{C}$ in $\mathcal{M}$ is defined by:

$$\begin{aligned} \tau &::= t \mid \{t\} \mid [l_1 : t_1, \ldots, l_n : t_n] \\ t &::= b \mid C \end{aligned}$$

In $\mathcal{M}$, equality on objects is defined to be oid equality. In addition, the equality relation in $\mathcal{M}$ is simply identity equality.

In $\mathcal{M}$, the definitions of database schemas and instances are similar to Definition 2.2 and 2.3, respectively. However, the signature determined by a schema Δ in $\mathcal{M}$ is defined to be a triple

$$\sigma(\Delta) = (r, E(\Delta), R(\Delta)),$$

where r, $E(\Delta)$ and $R(\Delta)$ are defined in the same way as in Definition 2.4 and 2.5.

Because the equality in $\mathcal{M}$ has a simple semantics, a schema Δ in $\mathcal{M}$ can be characterized by the type constraint $\Phi(\Delta)$ alone, which is defined in the same way as in Definition 2.6. As a result, an abstract database of Δ is defined to be a finite $\sigma(\Delta)$-structure satisfying $\Phi(\Delta)$, and $\mathcal{U}(\Delta)$ is defined to be the set of $\sigma(\Delta)$-structures which satisfy $\Phi(\Delta)$ and respect the finite set rule.

3 Word constraints in $\mathcal{M}^{\approx}$

In this section, we formally define word constraints in the context of $\mathcal{M}^{\approx}$. We first present the notion of paths. We then define word constraints and their associated implication and finite implication problems. Finally, we justify the abstraction of databases of $\mathcal{M}^{\approx}$ given in the last section with respect to word constraint implication.

3.1 Paths

We begin with a description of paths in terms of finite state automata [14].

Definition 3.1: Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$. The *finite state automata determined by* Δ is defined to be $M(\Delta) = (S, A, \delta, s, F)$, where

- the set of *states* S is $T(\Delta)$, the set of types determined by Δ;
- the *alphabet* A is $E(\Delta)$, the set of binary relation symbols determined by Δ;
- the *initial state* s is $DBtype$;
- the set of *final states* F is also $T(\Delta)$; and
- the *transition function* δ is defined as follows: For every $\tau \in T(\Delta)$,
  - if $\tau = \{\tau'\}$, then $\delta(\tau, *) = \tau'$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$, then for every $i \in [1, n]$, $\delta(\tau, l_i) = \tau_i$.

Define a (partial) function $\widehat{\delta} : T(\Delta) \times E(\Delta)^* \rightarrow T(\Delta)$ by:

$$\begin{aligned} \widehat{\delta}(\tau, \epsilon) &= \tau \\ \widehat{\delta}(\tau, \alpha K) &= \delta(\widehat{\delta}(\tau, \alpha), K) \end{aligned}$$

A *path* α *over* Δ is an element of $E(\Delta)^*$ such that there is $\tau \in T(\Delta)$ and $\widehat{\delta}(\tau, \alpha)$ is defined.

Let $Paths(\Delta)$ be the language accepted by $M(\Delta)$. An *initial path over* Δ is an element of $Paths(\Delta)$. The *type of an initial path* α, $type(\alpha)$, is defined to be $\widehat{\delta}(DBtype, \alpha)$. ■

It is easy to verify that $\widehat{\delta}$ is indeed a function. As a result, the type of an initial path is well-defined. In particular, the empty path ϵ is in $Paths(\Delta)$ and $type(\epsilon) = DBtype$.

Example 3.1: The finite state automata determined by the schema given in Example 2.2 is shown in Figure 2. ■

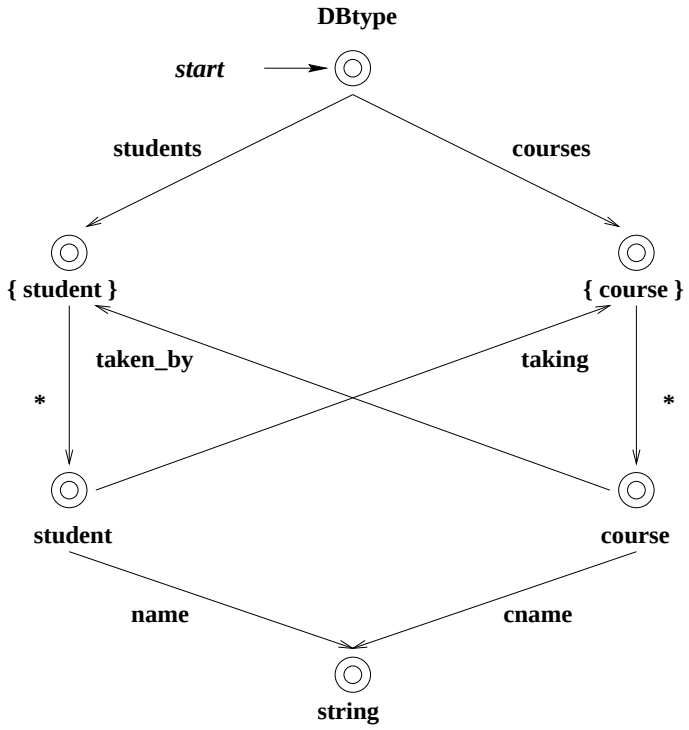

Figure 2: The finite state automata determined by the schema given in Example 2.2

Following [9, 10], we represent path α over Δ as a logic formula $\alpha(x, y)$, where x and y denote the tail and head nodes of the path, respectively. More precisely, $\alpha(x, y)$ is defined by:

$$\alpha(x,y) = \begin{cases} x = y & \text{if } \alpha = \epsilon \\ \exists z(\beta(x,z) \wedge *(z,y)) & \text{if } \alpha = \beta * \\ \exists z(\beta(x,z) \wedge l(z,y)) & \text{if } \alpha = \beta\, l \end{cases}$$

Here $\beta(x, z)$ is a formula representing the path β. We write $\alpha(x, y)$ as α when x and y are understood from the context.

In the sequel, we assume that all the paths over Δ are in the form of the formulas defined above.

The *concatenation* of paths $\alpha(x, z)$ and $\beta(z, y)$, denoted $\alpha(x, z) \cdot \beta(z, y)$ or simply $\alpha \cdot \beta$, is defined by:

$$\alpha(x,z) \cdot \beta(z,y) \;=\; \begin{cases} \beta(x,y) & \text{if } \alpha = \epsilon \\ \alpha'(x,u) \cdot \exists z(*(u,z) \wedge \beta(z,y)) & \text{if } \alpha(x,z) = \exists u(\alpha'(x,u) \wedge *(u,z)) \\ \alpha'(x,u) \cdot \exists z(l(u,z) \wedge \beta(z,y)) & \text{if } \alpha(x,z) = \exists u(\alpha'(x,u) \wedge l(u,z)) \end{cases}$$

The *length* of path α, $|\alpha|$, is defined by:

$$|\alpha| \;=\; \begin{cases} 0 & \text{if } \alpha = \epsilon \\ 1 + |\beta| & \text{if } \alpha = \beta \cdot * \\ 1 + |\beta| & \text{if } \alpha = \beta \cdot l \end{cases}$$

Definition 3.2: Let Δ be a schema in $\mathcal{M}^{\approx}$, α and β be paths over Δ. Then α is said to be a *prefix* of β, denoted $\alpha \preceq \beta$, if there exists a path γ over Δ, such that $\beta = \alpha \cdot \gamma$. ■

An important property of $M(\Delta)$ is described as follows.

Lemma 3.1: Let Δ be a schema in $\mathcal{M}^{\approx}$ and α be a path in $Paths(\Delta)$. Then for every $\sigma(\Delta)$-structure G satisfying $\Phi(\Delta)$ and for every node o in G, if $G \models \alpha(r^G, o)$, then $o \in R^G_\tau$, where $\tau = type(\alpha)$. ■

Proof: A straightforward induction on $|\alpha|$. ■

3.2 The definition of word constraints

Using the notion of path formulas, we define word constraints as follows.

Definition 3.3: A *word constraint φ over a schema Δ in $\mathcal{M}^{\approx}$* is a sentence of the form

$$\forall x\ (\alpha(r, x) \rightarrow \beta(r, x)),$$

where $\alpha, \beta \in Paths(\Delta)$ and $type(\alpha) = type(\beta)$. We denote α, β as $lt(\varphi)$ and $rt(\varphi)$, respectively.

We denote the set of all word constraints over schema Δ as $P_w(\Delta)$. ■

Obviously, $P_w(\Delta)$ is a language over vocabulary $\sigma(\Delta)$.

Example 3.2: The following are word constraints over the schema given in Example 2.2.

$$\begin{aligned} \phi &= \forall\, x\, (students \cdot * \cdot taking \cdot *(r, x) \rightarrow courses \cdot *(r, x)), \\ \varphi &= \forall\, x\, (courses \cdot * \cdot taken\_by \cdot *(r, x) \rightarrow students \cdot *(r, x)). \end{aligned}$$

In an instance $(\pi,\ \mu,\ d)$ of the schema, ϕ and φ are interpreted as:

$$\begin{aligned} \forall\, x\, (\exists\, y\, (y \in d.students \wedge x \in y.taking) &\rightarrow x \in d.courses) \\ \forall\, x\, (\exists\, y\, (y \in d.courses \wedge x \in y.taken\_by) &\rightarrow x \in d.students) \end{aligned}$$

respectively. As mentioned earlier, $v.l$ stands for the projection of record v at attribute l. The constraint ϕ states: "any course taken by a student in database d is a course in d", and

φ states: "any student who is taking a course in database d is a student in d". These two constraints express referential integrity for the database. ■

We borrow the standard definitions of models and implication from first-order logic [13]. Let Δ be a schema in $\mathcal{M}^{\approx}$, G be a structure in $\mathcal{U}(\Delta)$ and φ be a constraint in $P_w(\Delta)$. Then we write $G \models \varphi$ if G is a model of φ. Given a finite subset Σ of $P_w(\Delta)$, we use $\Sigma \models_\Delta \varphi$ to denote that Σ *implies* φ. That is, for every structure $G \in \mathcal{U}(\Delta)$, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_{(f,\,\Delta)} \varphi$ to denote that Σ *finitely implies* φ. That is, for every structure $G \in \mathcal{U}_f(\Delta)$, if $G \models \Sigma$, then $G \models \varphi$. We write $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\,\Delta)} \varphi$) as $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) if Δ is understood from the context.

Definition 3.4: Let Δ be a schema in $\mathcal{M}^{\approx}$. The *(finite) implication problem for word constraints ($P_w(\Delta)$) over* Δ is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, whether $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\,\Delta)} \varphi$). ■

3.3 Justification of the abstraction

Next, we justify the abstraction of databases of $\mathcal{M}^{\approx}$ defined in the last section.

As illustrated by Example 3.2, word constraints over a schema Δ can be naturally interpreted in database instances of Δ. Likewise, the notion "$I \models \varphi$" can also be defined for an instance I of Δ and a constraint φ of $P_w(\Delta)$.

The agreement between databases and their abstraction with respect to word constraint implication is revealed by the following lemma.

Lemma 3.2: Let Δ be any schema in $\mathcal{M}^{\approx}$. Then for each $I \in \mathcal{I}(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$, such that

$$\text{for any } \varphi \in P_w(\Delta),\ \ I \models \varphi \ \text{ iff } \ G \models \varphi \tag{1}$$

Similarly, for each $G \in \mathcal{U}_f(\Delta)$, there is $I \in \mathcal{I}(\Delta)$, such that (1) holds. ■

Proof: Let $\Delta = (\mathcal{C},\, \nu,\, DBtype)$.

(1) Given $I \in \mathcal{I}(\Delta)$, we construct $G \in \mathcal{U}_f(\Delta)$, such that for each $\varphi \in P_w(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

Let $I = (\pi,\, \mu,\, d)$. Then we define V to be the smallest set satisfying the following:

1. $d \in V$;
2. for every $v \in V$,
   - if v is a set, then every element of v is in V;
   - if v is a record (or v is an object and $\mu(v)$ is a record), then every attribute of v (or $\mu(v)$) is in V.

For every $v \in V$, let $o(v)$ be a distinct node. Let $G = (|G|,\, r^G,\, E^G,\, R^G,\, Q^G)$, where

- $|G| = \{o(v) \mid v \in V\}$;
- $r^G = o(d)$;
- for each $o(v) \in |G|$ and $\tau \in T(\Delta)$, $G \models R^G_\tau(o(v))$ iff v is of type τ;
- for each $\tau \in T(\Delta)$, the relation $\approx_\tau$ is defined to be $\{(o(v), o(v)) \mid o(v) \in R^G_\tau\}$;
- for all $o(v), o(v') \in |G|$,
  - for each $l \in \mathcal{L} \cap E(\Delta)$, $G \models l(o(v), o(v'))$ iff $v' = v.l$ (or $v' = \mu(v).l$ if v is an object);
  - $G \models *(o(v), o(v'))$ iff $v' \in v$.

Then it is straightforward to verify the following:

- $G \in \mathcal{U}_f(\Delta)$; that is, G is a finite $\sigma(\Delta)$-structure and $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$;
- for each $\varphi \in P_w(\Delta)$, $G \models \varphi$ iff $I \models \varphi$. This can be easily verified by *reductio*.

(2) Given $G = (|G|, r^G, E^G, R^G, Q^G)$ in $\mathcal{U}_f(\Delta)$, we define $I = (\pi, \mu, d)$ in $\mathcal{I}(\Delta)$, such that for every $\varphi \in P_w(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

To simplify the discussion, we assume that for every atomic type b, its domain D_b is infinite (a similar proof for the finite case can be found in [9]). By this assumption, there exists an injective mapping $g_b : R^G_b \to D_b$, where R^G_b is the unary relation in G denoting the sort b.

For every $C \in \mathcal{C}$, let $\pi(C) = R^G_C$. We then define a mapping $f : |G| \to \bigcup_{\tau \in T(\Delta)} [\![\tau]\!]_\pi$ as follows: For each $o \in |G|$,

- if $o \in R^G_C$ for some $C \in \mathcal{C}$, then let $f(o) = o$;
- if $o \in R^G_b$ for some atomic type b, then let $f(o) = g(o)$;
- if $o \in R^G_\tau$ and $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then let $f(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)]$, where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$;
- if $o \in R^G_\tau$ and $\tau = \{\tau'\}$, then let $f(o) = \{f(o') \mid o' \in |G|,\ G \models *(o, o')\}$.

Note that f is well-defined and is an injection, since G is finite and $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$. Now let

- $d = f(r^G)$;
- for each $C \in \mathcal{C}$ and each $o \in \pi(C)$, if $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then

  $$\mu(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)],$$

  where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$.

Again, this is well-defined. In addition, it is easy to verify that $I \in \mathcal{I}(\Delta)$, and $G \models \varphi$ iff $I \models \varphi$. ■

From Lemma 3.2 follows immediately the corollary below.

Corollary 3.3: Let Δ be any schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be any finite subset of $P_w(\Delta)$. Then there is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$ if and only if there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$. ■

4 Word constraint implication in $\mathcal{M}^{\approx}$

In this section, we establish the decidability of the implication and finite implication problems for word constraints in the context of $\mathcal{M}^{\approx}$.

Theorem 4.1: Over arbitrary schema Δ in $\mathcal{M}^{\approx}$, both the implication and finite implication problems for $P_w(\Delta)$ are decidable. ■

We prove Theorem 4.1 by giving a small model argument. Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$. Given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model G of size at most 2^{mN}, and $G \in \mathcal{U}_f(\Delta)$, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the *maximum record width of* Δ defined by

$$m = 1 + max\{n \mid \tau \in T(\Delta), \tau = [l_1 : \tau_1, \ \ldots, \ l_n : \tau_n] \text{ or } \nu(\tau) = [l_1 : \tau_1, \ \ldots, \ l_n : \tau_n] \text{ if } \tau \in \mathcal{C}\}.$$

Before we give the proof, we first describe two techniques used to establish the small model property. The first is a simple filtration argument. Using it we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure G such that the size of G is at most 2^{mN} and

$$G \models \Sigma \wedge \neg\varphi \wedge \Phi(\Delta).$$

It should be noted that if the equality constraint $\Phi^{\approx}(\Delta)$ is not taken into account, this filtration argument alone suffices. The second technique is referred to as *identifying* operation. Using it we construct H from G such that the size of H is no larger than the size of G, and in addition, under certain conditions,

$$H \models \Sigma \wedge \neg\varphi \wedge \Phi(\Delta) \wedge \Phi^{\approx}(\Delta).$$

Finally, we use a slightly stronger filtration argument and the identifying operation to prove Theorem 4.1.

4.1 A filtration argument

We first present a simple filtration argument. It should be noted that for an object-oriented model which does not support complex values with nested structures and is in the flavor of

the nested relational model, this argument alone is sufficient to establish the small model property for word constraint implication.

We begin with a few definitions.

Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. Then we define the following:

$$\begin{aligned} Pts(\Sigma \cup \{\varphi\}) &= \{lt(\phi), rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \\ CloPts(\Sigma \cup \{\varphi\}) &= \{\rho \mid \varrho \in Pts(\Sigma \cup \{\varphi\}),\ \rho \preceq \varrho\} \end{aligned}$$

It is straightforward to verify the following.

Lemma 4.2: Let N be the length of $\bigwedge \Sigma \wedge \neg\varphi$. Then the cardinality of $CloPts(\Sigma \cup \{\varphi\})$ is at most N. ∎

Let G be a $\sigma(\Delta)$-structure. Then for every $a \in |G|$, we define

$$lb(a,\, G,\, \Sigma \cup \{\varphi\}) = \{\rho \mid \rho \in CloPts(\Sigma \cup \{\varphi\}),\ G \models \rho(r^G,\, a)\}.$$

In addition, we define *the label of G with respect to* $\Sigma \cup \{\varphi\}$ to be

$$LB(G,\, \Sigma \cup \{\varphi\}) = \{lb(a,\, G,\, \Sigma \cup \{\varphi\}) \mid a \in |G|\}.$$

An important property of $LB(G,\, \Sigma \cup \{\varphi\})$ is that it characterizes whether $G \models \bigwedge \Sigma \wedge \neg\varphi$.

Lemma 4.3: Suppose that $\bigwedge \Sigma \wedge \neg\varphi$ has a model G. Then for every $\sigma(\Delta)$-structure H, if $LB(H,\, \Sigma \cup \{\varphi\}) = LB(G,\, \Sigma \cup \{\varphi\})$, then $H \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Proof: We first show that $H \models \Sigma$. Suppose, for *reductio*, that there is $\phi \in \Sigma$ and $a \in |H|$, such that

$$H \models lt(\phi)(r^H,\, a) \wedge \neg rt(\phi)(r^H,\, a).$$

Then we have $lt(\phi) \in lb(a,\, H,\, \Sigma \cup \{\varphi\})$, but $rt(\phi) \notin lb(a,\, H,\, \Sigma \cup \{\varphi\})$. By the assumption that $LB(H,\, \Sigma \cup \{\varphi\}) = LB(G,\, \Sigma \cup \{\varphi\})$, we have

$$lb(a,\, H,\, \Sigma \cup \{\varphi\}) \in LB(G,\, \Sigma \cup \{\varphi\}).$$

Hence there is $b \in |G|$ such that

$$lb(b,\, G,\, \Sigma \cup \{\varphi\}) = lb(a,\, H,\, \Sigma \cup \{\varphi\}).$$

Therefore,

$$G \models lt(\phi)(r^G,\, b) \wedge \neg rt(\phi)(r^G,\, b).$$

Hence $G \not\models \phi$. This contradicts the assumption that $G \models \Sigma$.

Next, we show that $H \models \neg\varphi$. By $G \models \neg\varphi$, there exists $b \in |G|$ such that

$$G \models lt(\varphi)(r^G,\, b) \wedge \neg rt(\varphi)(r^G,\, b).$$

Hence we have $lt(\varphi) \in lb(b, G, \Sigma \cup \{\varphi\})$, but $rt(\varphi) \notin lb(b, G, \Sigma \cup \{\varphi\})$. By the assumption that $LB(H, \Sigma \cup \{\varphi\}) = LB(G, \Sigma \cup \{\varphi\})$, there is $a \in |H|$ such that

$$lb(a, H, \Sigma \cup \{\varphi\}) = lb(b, G, \Sigma \cup \{\varphi\}).$$

Therefore,

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

Hence $H \models \neg\varphi$. ■

Next, we present the filtration argument.

Proposition 4.4: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model G and $G \models \Phi(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

and the size of H is at most 2^{mN}, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ■

Proof: Let $\Delta = (\mathcal{C}, \nu, DBtype)$. Since $G \models \Phi(\Delta)$, for every $a \in |G|$, there is $\tau \in T(\Delta)$ such that $a \in R^G_\tau$. We define $mlb(a)$ to be either

- $lb(a, G, \Sigma \cup \{\varphi\})$, if $\tau \notin \mathcal{C}$ and τ is not a record type, or
- $(lb(a, G, \Sigma \cup \{\varphi\}), (l_1, lb(a_1, G, \Sigma \cup \{\varphi\})), \ldots, (l_n, lb(a_n, G, \Sigma \cup \{\varphi\})))$, if τ is record type $[l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $\tau \in \mathcal{C}$ and $\nu(\tau) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and for $i \in [1, n]$, $G \models l_i(a, a_i)$.

Let $MLB(G) = \{mlb(a) \mid a \in |G|\}$. For each $s \in MLB(G)$, let o_s be a distinct node. We define a function $f : |G| \to MLB(G)$ such that for each $a \in |G|$, $f : a \mapsto o_s$ where $s = mlb(a)$.

Using f, we define $H = (|H|, r^H, E^H, R^H, Q^H)$ as follows.

- $|H| = \{f(a) \mid a \in |G|\}$.
- $r^H = f(r^G)$.
- E^H is populated as follows: for each $K \in E$ and $o_a, o_b \in |H|$, $H \models K(o_a, o_b)$ iff there exist $a, b \in |G|$ such that $G \models K(a, b)$, $f(a) = o_a$ and $f(b) = o_b$.
- For every $\tau \in T(\Delta)$ and $o \in |H|$, $o \in R^H_\tau$ iff there is $a \in |G|$ such that $f(a) = o$ and $a \in R^G_\tau$. This is well-defined by Lemma 3.1 and the assumption that $G \models \Phi(\Delta)$.
- For every $\tau \in T(\Delta)$, we define $\approx_\tau$ be $\{(o, o) \mid o \in R^H_\tau\}$.

Next, we show that H is indeed the structure described in the proposition.

(1) The size of H.

For every $a \in |G|$, either $mlb(a) \subseteq CloPts(\Sigma \cup \{\varphi\})$, or $mlb(a) \subseteq CloPts^m(\Sigma \cup \{\varphi\})$. Hence by Lemma 4.2, the size of H is at most 2^{mN}.

(2) $H \models \Phi(\Delta)$.

By Lemma 3.1 and the assumption that $G \models \Phi(\Delta)$, it is easy to verify that for every $o \in |H|$, there is a unique $\tau \in T(\Delta)$ such that $o \in R^H_\tau$. In addition, by the definition of H, it is easy to verify the following.

- $r^H \in R^H_{DBtype}$.
- For every $\tau \in T(D)$ and $o \in R^H_\tau$,
  - if τ is either a base type or a set type, then $H \models \phi_\tau(o)$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$, then

$$H \models \forall\, y\,(\bigwedge_{l \in E(\Delta)\setminus\{l_1,\ldots,l_n\}} \neg l(o,y)) \ \wedge \bigwedge_{i\in[1,n]} (\exists\, y\, l_i(o,y) \ \wedge \forall\, y\,(l_i(o,y) \rightarrow R_{\tau_i}(y))).$$

By the definition of $mlb(o)$, it can also be shown that if τ or $\nu(\tau)$ is a record type, then

$$H \models \bigwedge_{i\in[1,n]} \exists!\, y\, l_i(o,y).$$

To see this, without loss of generality, suppose for *reductio* that there exist $o_1, o_2 \in |H|$, such that

$$H \models l_1(o,\, o_1) \wedge l_1(o,\, o_2) \wedge o_1 \neq o_2.$$

Then by the definition of E^H, there exist $a, b, c, d \in |G|$ such that $f(a) = f(c) = o$, $f(b) = o_1$, $f(d) = o_2$, and $G \models l_1(a,\, b) \wedge l_1(c,\, d)$. By the definition of f and mlb, we have:

$$\begin{array}{rcl} mlb(a) & = & (lb(a,\, G,\, \Sigma \cup \{\varphi\}),\ (l_1,\, lb(b,\, G,\, \Sigma \cup \{\varphi\})),\ \ldots\,) \\ mlb(c) & = & (lb(c,\, G,\, \Sigma \cup \{\varphi\}),\ (l_1,\, lb(d,\, G,\, \Sigma \cup \{\varphi\})),\ \ldots\,) \\ mlb(a) & = & mlb(c) \\ mlb(b) & \neq & mlb(d) \end{array}$$

By Definition 2.1, τ_1 is neither a class type nor a record type. Hence

$$\begin{array}{rcl} mlb(b) & = & lb(b,\, G,\, \Sigma \cup \{\varphi\}), \\ mlb(d) & = & lb(d,\, G,\, \Sigma \cup \{\varphi\}). \end{array}$$

Hence by $mlb(b) \neq mlb(d)$, we have $mlb(a) \neq mlb(c)$. This contradicts the assumption that $f(a) = f(c)$.

Therefore, we have $H \models \Phi(\Delta)$.

(2) $H \models \bigwedge \Sigma \wedge \neg\varphi$.

It suffices to show the following claim.

Claim: For every $a \in |G|$, $lb(a, G, \Sigma \cup \{\varphi\}) = lb(f(a), H, \Sigma \cup \{\varphi\})$.

For if the claim holds, then $LB(H, \Sigma \cup \{\varphi\}) = LB(G, \Sigma \cup \{\varphi\})$. Thus by Lemma 4.3, we have $H \models \bigwedge \Sigma \wedge \neg\varphi$.

Next, we show that for every $\alpha \in CloPts(\Sigma \cup \{\varphi\})$,

$$\alpha \in lb(a, G, \Sigma \cup \{\varphi\}) \quad \text{iff} \quad \alpha \in lb(f(a), H, \Sigma \cup \{\varphi\}).$$

This can be verified by induction on $|\alpha|$ as follows.

Base case: $\alpha = \epsilon$. Note that r^G is the unique node in G such that $\epsilon \in lb(r^G, G, \Sigma \cup \{\varphi\})$. Hence $\epsilon \in lb(a, G, \Sigma \cup \{\varphi\})$ iff $a = r^G$ iff $f(a) = r^H$ iff $\epsilon \in lb(r^H, H, \Sigma \cup \{\varphi\})$.

Inductive step: Assume the claim for $|\alpha|$. We next show that the claim also holds for $\alpha \cdot K$, where $\alpha \cdot K \in CloPts(\Sigma \cup \{\varphi\})$.

If $\alpha \cdot K \in lb(a, G, \Sigma \cup \{\varphi\})$, then there is $b \in |G|$ such that

$$G \models \alpha(r^G, b) \wedge K(b, a).$$

By induction hypothesis, we have

$$\alpha \in lb(f(b), H, \Sigma \cup \{\varphi\}).$$

By the definition of E^H, we have

$$H \models K(f(b), f(a)).$$

Therefore, $\alpha \cdot K \in lb(f(a), H, \Sigma \cup \{\varphi\})$.

If $\alpha \cdot K \in lb(f(a), H, \Sigma \cup \{\varphi\})$, then there exists $b \in |H|$ such that

$$H \models \alpha(r^H, b) \wedge K(b, f(a)).$$

By the definition of E^H, there exist $o_1, o_2 \in |G|$ such that $f(o_1) = b$, $f(o_2) = f(a)$ and $G \models K(o_1, o_2)$. By induction hypothesis, we have

$$\alpha \in lb(o_1, G, \Sigma \cup \{\varphi\}).$$

Hence $\alpha \cdot K \in lb(o_2, G, \Sigma \cup \{\varphi\})$. By $f(o_2) = f(a)$ and the definition of f, we have

$$lb(a, G, \Sigma \cup \{\varphi\}) = lb(o_2, G, \Sigma \cup \{\varphi\}).$$

Hence $\alpha \cdot K \in lb(a, G, \Sigma \cup \{\varphi\})$.

Therefore, the claim holds.

This completes the proof of Proposition 4.4 ∎

4.2 Identifying operation

Next, we introduce an operation called *identifying*. Using this operation we are able to construct structures that satisfy necessary equality constraints. More specifically, let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. Then if there is a $\sigma(\Delta)$-structure G such that

$$G \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$$

and G satisfies certain conditions, then we can construct H from G such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta) \wedge \Phi^{\approx}(\Delta),$$

and the size of H is no larger than the size of G.

Before we define the identifying operation, we first present some basic properties of the data model $\mathcal{M}^{\approx}$.

Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Delta = (\mathcal{C}, \nu, DBtype)$. Then we use $BC(\Delta)$ to denote the set $T(\Delta) \cap (\mathcal{B} \cup \mathcal{C})$.

Lemma 4.5: Let Δ be a schema in $\mathcal{M}^{\approx}$, $\Delta = (\mathcal{C}, \nu, DBtype)$ and G be a $\sigma(\Delta)$-structure such that $G \models \Phi(\Delta)$. Then G has the following properties.

1. For every $a \in |G|$, if $a \in R^G_\tau$ and $\tau \notin BC(\Delta)$, then either $\tau = DBtype$ or $\tau = \{\tau'\}$.
2. For every $a \in |G|$, if $a \in R^G_{DBtype}$ and $a \neq r^G$, then for every path $\alpha \in Paths(\Delta)$, $G \not\models \alpha(r^G, a)$.

■

Proof: By Definition 2.4 and 3.1, it is easy to see that if $\tau \in T(\Delta)$, then there is $\alpha \in Paths(\Delta)$ such that $type(\alpha) = \tau$. Hence it suffices to show the following claim.

Claim: For every $\alpha \in Paths(\Delta)$, if $\alpha \neq \epsilon$, then either $type(\alpha) \in BC(\Delta)$ or $type(\alpha)$ is a set type.

For if the claim holds, then for every $\tau \in T(\Delta) \setminus BC(\Delta)$, τ is either $DBtype$ or a set type. That is, the first statement of the lemma holds. The second statement can also be verified by *reductio*, using the claim. Suppose, for *reductio*, that there is $a \in R^G_{DBtype}$ and $\alpha \in Paths(\Delta)$ such that $a \neq r^G$ and $G \models \alpha(r^G, a)$. Then $\alpha \neq \epsilon$ since $a \neq r^G$. If α is not ϵ, then it follows from the claim that $type(\alpha)$ cannot be $DBtype$. However, if $G \models \Phi(\Delta)$ and $G \models \alpha(r^G, a)$, then by Lemma 3.1 and $a \in R^G_{DBtype}$, we have $type(\alpha) = DBtype$. This contradicts the claim.

Next, we show the claim by induction on $|\alpha|$.

Base case: $\alpha = K$ for some $K \in E$. By Definition 2.2, $type(K)$ must be a set type. Hence the claim holds in this case.

Inductive step: Assume the claim for $|\alpha|$. We show that the claim also holds for $\alpha \cdot K$, where $\alpha \cdot K \in Paths(\Delta)$. By induction hypothesis, $type(\alpha)$ is either in $BC(\Delta)$ or is a set type. Since $\alpha \cdot K \in Paths(\Delta)$, $type(\alpha) \notin \mathcal{B}$. By Definition 2.2 and 2.4, if $type(\alpha) \in \mathcal{C}$ then $type(\alpha \cdot K)$ is

either in $BC(\Delta)$ or is a set type. Similarly, if $type(\alpha)$ is a set type, then $type(\alpha \cdot K) \in BC(\Delta)$. Hence the claim holds for $\alpha \cdot K$. ∎

Corollary 4.6: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If there is G in $\mathcal{U}(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$, then there is H in $\mathcal{U}(\Delta)$ such that $R^H_{DBtype} = \{r^H\}$, $H \models \bigwedge \Sigma \wedge \neg\varphi$, and the size of H is no larger than the size of G. ∎

Proof: Let H be the substructure of G such that

$$|H| = \{a \mid a \in |G|,\ G \models \alpha(r^G,\ a) \text{ for some } \alpha \in Paths(\Delta)\}.$$

It is easy to verify that $H \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$ and

$$LB(H,\ \Sigma \cup \{\varphi\}) \ = \ LB(G,\ \Sigma \cup \{\varphi\}).$$

Hence by Lemma 4.3, we have $H \models \bigwedge \Sigma \wedge \neg\varphi$. By Lemma 4.5, we also have $R^H_{DBtype} = \{r^H\}$. ∎

In the sequel we assume that for every $\sigma(\Delta)$-structure G, $R^G_{DBtype} = \{r^G\}$. By Corollary 4.6, this assumption does not affect the outcome of word constraint implication in $\mathcal{M}^{\approx}$.

Next, we define the identifying operation.

Definition 4.1: Let Δ be a schema in $\mathcal{M}^{\approx}$, G be a $\sigma(\Delta)$-structure that satisfies $\Phi(\Delta)$, $G = (|G|,\ r^G,\ E^G,\ R^G,\ Q^G)$, and o_1, o_2 be two distinct nodes in $|G|$. Then o_1 and o_2 are said to be *identifiable* if there is $\tau \in T(\Delta) \setminus BC(\Delta)$, such that $G \models R^G_\tau(o_1) \wedge R^G_\tau(o_2)$, and in addition, the following conditions are satisfied.

- If $\tau = \{\tau'\}$, then for every $a \in |G|$,

$$G \models *(o_1,\ a) \quad \text{iff} \quad G \models *(o_2,\ a).$$

- If $\tau = [l_1 : \tau_1,\ \ldots,\ l_n : \tau_n]$, then for every $i \in [1, n]$ and $a \in |G|$,

$$G \models l_i(o_1,\ a) \quad \text{iff} \quad G \models l_i(o_2,\ a).$$

The *structure resulting from identifying* o_1 *and* o_2 *in* G is defined to be

$$G_1 = (|G_1|,\ r^{G_1},\ E^{G_1},\ R^{G_1},\ Q^{G_1}),$$

where

- $|G_1| = (|G| \setminus \{o_1, o_2\}) \cup \{o\}$, where $o \notin |G|$,
- $r^{G_1} = r^G$,
- for all $a, b \in |G_1|$ and $K \in E(\Delta)$,

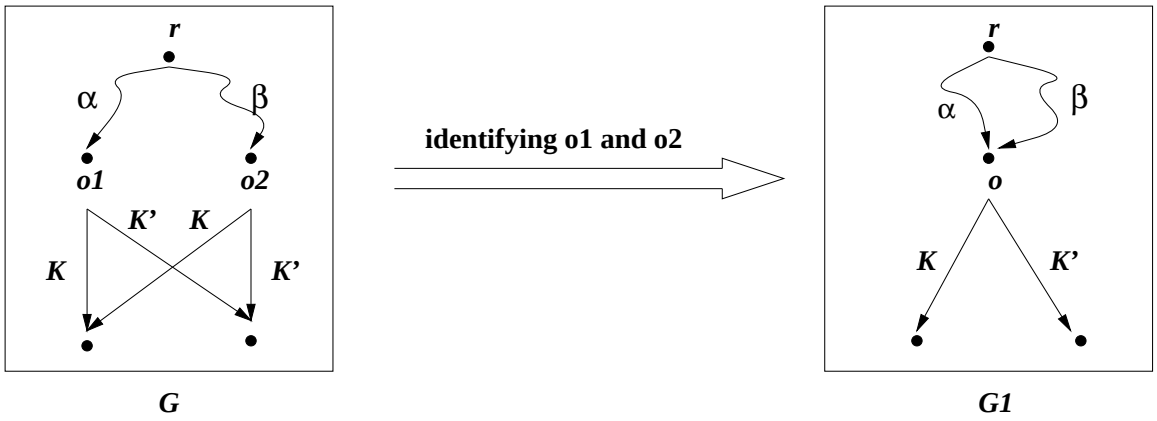

Figure 3: The identifying operation

  - if $a \neq o$ and $b \neq o$, then $G_1 \models K(a,b)$ iff $G \models K(a,b)$,
  - if $a = o$ and $b \neq o$, then $G_1 \models K(o,b)$ iff $G \models K(o_1,b)$ (iff $G \models K(o_2,b)$),
  - if $a \neq o$ and $b = o$, then $G_1 \models K(a,o)$ iff $G \models K(a,o_1) \vee K(a,o_2)$;

- for every $\tau' \in T(\Delta) \setminus \{\tau\}$, $R^{G_1}_{\tau'} = R^G_{\tau'}$; and in addition,

$$R^{G_1}_{\tau} = R^G_{\tau} \cup \{o\} \setminus \{o_1, o_2\};$$

- for every $\tau' \in T(\Delta) \setminus \{\tau\}$ and all $a, b \in |G_1|$, $a \approx^{G_1}_{\tau'} b$ iff $a \approx^G_{\tau'} b$; and in addition,

$$a \approx^{G_1}_{\tau} b \quad \text{iff} \quad (a,b) \text{ is in } \approx^G_{\tau} [o_1/o, o_2/o] \cup \{(o,\, o)\},$$

  where $\approx^G_{\tau} [o_1/o, o_2/o]$ stands for substituting o for every occurrence of o_1 and o_2 in the relation $\approx^G_{\tau}$.

■

The identifying operation is shown in Figure 3.

The following should be noted about structure G_1 described in Definition 4.1.

- The root r^G is not in $\{o_1, o_2\}$. This is because we assume $R^G_{DBtype} = \{r^G\}$ by Corollary 4.6.

- There is no $K \in E(\Delta)$ such that $G_1 \models K(o,o)$. This is because τ is not a class type. As a result, τ cannot be defined in terms of itself. Thus $G \not\models K(o_1,o_2) \vee K(o_2,o_1)$, since $G \models \Phi(\Delta)$.

- For every $a \in |G|$ and $K \in E(\Delta)$, $G_1 \models K(o,a)$ iff $G \models K(o_2,a)$. This is because $G \models K(o_1,a)$ iff $G \models K(o_2,a)$.

Next, we show that the identifying operation has certain properties.

Lemma 4.7: Let Δ, G, o_1, o_2, G_1 and o be as described in Definition 4.1. Then G_1 has the following properties.

1. $G_1 \models \Phi(\Delta)$.
2. For every $\alpha \in Paths(\Delta)$ and $a \in |G_1|$,
   (a) if $a \neq o$, then $a \in |G|$, and in addition, $G_1 \models \alpha(r^{G_1}, a)$ iff $G \models \alpha(r^G, a)$;
   (b) if $a = o$, then $G_1 \models \alpha(r^{G_1}, o)$ iff $G \models \alpha(r^G, o_1) \vee \alpha(r^G, o_2)$.

■

Proof: The first statement of the lemma can be easily verified by *reductio*. We prove the second statement by induction on $|\alpha|$.

Base case: $\alpha = \epsilon$. Note that $r^G \notin \{o_1, o_2\}$. Hence $G_1 \models \epsilon(r^{G_1}, a)$ iff $a = r^{G_1}$ iff $a = r^G$ iff $G \models \epsilon(r^G, a)$. Hence the statement holds for this case.

Inductive step: Assume the statement for $|\alpha|$. We show that the statement also holds for $\alpha \cdot K$, where $\alpha \cdot K \in Paths(\Delta)$.

(1) Assume that $a \neq o$. By Definition 4.1, clearly $a \in |G|$.

First, assume that $G \models \alpha \cdot K(r^G, a)$. Then there exists $b \in |G|$, such that

$$G \models \alpha(r^G, b) \wedge K(b, a).$$

If $b \notin \{o_1, o_2\}$, then by Definition 4.1, $b \in |G_1|$ and $b \neq o$. Thus by induction hypothesis and Definition 4.1, we have $G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a)$.

If $b \in \{o_1, o_2\}$, then by induction hypothesis we have

$$G_1 \models \alpha(r^{G_1}, o).$$

By Definition 4.1, we have

$$G_1 \models K(o, a).$$

Hence $G_1 \models \alpha \cdot K(a, b)$.

Conversely, assume that $G_1 \models \alpha \cdot K(r^{G_1}, a)$. Then there exists $b \in |G_1|$ such that

$$G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a).$$

If $b \neq o$, then by induction hypothesis, $b \in |G|$ and

$$G \models \alpha(r^G, b).$$

By Definition 4.1, we have

$$G \models K(b, a).$$

Hence $G \models \alpha(r^G, b) \wedge K(b, a)$.

If $b = o$, then by induction hypothesis,

$$G \models \alpha(r^G, o_1) \vee \alpha(r^G, o_2).$$

By Definition 4.1, we have

$$G \models K(o_1, a) \wedge K(o_2, a).$$

Hence $G \models \alpha(r^G, b) \wedge K(b, a)$.

(2) Assume that $a = o$.

First, assume that $G \models \alpha \cdot K(r^G, o_1) \vee \alpha \cdot K(r^G, o_2)$. Without loss of generality, assume that $G \models \alpha \cdot K(r^G, o_1)$. Then there exists $b \in |G|$, such that

$$G \models \alpha(r^G, b) \wedge K(b, o_1).$$

Clearly, $b \notin \{o_1, o_2\}$, since otherwise we would have $G_1 \models K(o, o)$. By Definition 4.1, $b \in |G_1|$ and $b \neq o$. By induction hypothesis,

$$G_1 \models \alpha(r^{G_1}, b).$$

By Definition 4.1, we have

$$G_1 \models K(b, o).$$

Hence $G_1 \models \alpha \cdot K(r^{G_1}, a)$.

Conversely, assume that $G_1 \models \alpha \cdot K(r^{G_1}, a)$. Then there exists $b \in |G_1|$ such that

$$G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a).$$

Clearly $b \neq o$ since otherwise we would have $G_1 \models K(o, o)$. By induction hypothesis, $b \in |G|$ and

$$G \models \alpha(r^G, b).$$

By Definition 4.1,

$$G \models K(b, o_1) \vee K(b, o_2).$$

Hence $G \models \alpha \cdot K(r^G, o_1) \vee \alpha \cdot K(r^G, o_1)$.

This completes the proof of Lemma 4.7. ∎

Using Lemma 4.7, we study the impact of the identifying operation on word constraint implication. To do this, we first introduce the following notion.

Definition 4.2: Let Δ, G, o_1 and o_2 be as described in Definition 4.1. Let φ be a word constraint in $P_w(\Delta)$. Then G is said to *respect* $\neg\varphi$ *when identifying* o_1 *and* o_2 if there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a),$$

and in addition, either $a \notin \{o_1, o_2\}$, or $a = o_1$ and $G \models \neg rt(\varphi)(r^G, o_2)$. ∎

Corollary 4.8: Let Δ, G, o_1, o_2 and G_1 be as described in Definition 4.1. Let $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $G \models \Sigma$, then $G_1 \models \Sigma$. In addition, if G respects $\neg\varphi$ when identifying o_1 and o_2, then $G_1 \models \neg\varphi$. ∎

Proof: We first show that if $G \models \Sigma$, then $G_1 \models \Sigma$. Suppose, for *reductio*, that there exists $\phi \in \Sigma$ and $b \in |G_1|$, such that

$$G_1 \models lt(\phi)(r^{G_1}, b) \wedge \neg rt(\phi)(r^{G_1}, b).$$

If $b \neq o$, then by Lemma 4.7, $b \in |G|$ and in addition,

$$G \models lt(\phi)(r^G, b) \wedge \neg rt(\phi)(r^G, b).$$

This contradicts the assumption that $G \models \Sigma$.

If $b = o$, then by $G \models \phi$, we have

$$G \models \neg lt(\phi)(r^G, o_1) \vee rt(\phi)(r^G, o_1)$$

and

$$G \models \neg lt(\phi)(r^G, o_2) \vee rt(\phi)(r^G, o_2).$$

Again by Lemma 4.7, we have

$$G_1 \models \neg lt(\phi)(r^G, o) \vee rt(\phi)(r^G, o).$$

This contradicts the assumption that $G_1 \not\models \phi$.

Next, we show that if G respects $\neg\varphi$ when identifying o_1 and o_2, then $G_1 \not\models \varphi$. Let $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a).$$

If $a \notin \{o_1, o_2\}$, then $a \in |G_1|$ and $a \neq o$. Since $G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a)$, by Lemma 4.7, we have

$$G_1 \models lt(\varphi)(r^{G_1}, a) \wedge \neg rt(\varphi)(r^{G_1}, a).$$

That is, $G_1 \not\models \varphi$.

If $a = o_1$ and $G \models \neg rt(\varphi)(r^G, o_2)$, then again by Lemma 4.7, we have

$$G_1 \models lt(\varphi)(r^{G_1}, o) \wedge \neg rt(\varphi)(r^{G_1}, o).$$

Therefore, $G_1 \not\models \varphi$. ∎

Next, we illustrate how to construct structures that satisfy equality constraints using the identifying operation.

Definition 4.3: Let Δ be a schema in $\mathcal{M}^{\approx}$ and H be a finite $\sigma(\Delta)$-structure satisfying $\Phi(\Delta)$. Then the *ultimately identified structure constructed from* H is defined to be structure G_m, where

$$\begin{aligned} G_1 &= H, \\ G_{i+1} &= \text{the structure resulting from identifying two identifiable nodes in } G_i, \end{aligned}$$

and G_m is the structure constructed at stage m such that there are no distinct identifiable nodes o_1, o_2 in $|G_m|$. ■

Ultimately identified structures have the following properties.

Proposition 4.9: Let Δ and H be as described in Definition 4.3. Then the following statements hold.

1. The ultimately identified structure G constructed from H exists. In addition, the size of G is no larger than the size of H.
2. For every finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, if $H \models \Sigma$ then $G \models \Sigma$. In addition, if $H \models \neg\varphi$ and at each stage i of the construction described in Definition 4.3, G_i respects $\neg\varphi$ when identifying nodes, then $G \models \neg\varphi$.
3. $G \models \Phi(\Delta)$.
4. If for any $\tau \in T(\Delta)$, $H \models \forall\, x\, y\, (x \approx_\tau^H y \rightarrow x = y)$, then $G \models \Phi^{\approx}(\Delta)$.

■

Proof:

(1) To see that the ultimately identified structure G constructed from H exists, consider the sequence of structures constructed in Definition 4.3. This sequence is finite since H is finite and in addition, the size of G_{i+1} is strictly less than the size of G_i unless $i = m$. As a result, G exists. In addition, the size of G is no larger than the size of H.

(2) The second statement of the proposition can be verified by a straightforward induction on stage i, using Corollary 4.8.

(3) The third statement of the proposition follows from Lemma 4.7.

(4) To show the last statement of the proposition, first, by induction on i, it is easy to show that for any $\tau \in T(\Delta)$,

$$G_i \models \forall\, x\, y\, (x \approx_\tau^{G_i} y \rightarrow x = y).$$

Therefore,

$$G \models \forall\, x\, y\, (x \approx_\tau^{G} y \rightarrow x = y). \qquad (\dagger)$$

Second, we show that $G \models \bigwedge_{\tau \in T(\Delta)} \forall x\, y\, \phi^{\approx}_{\tau}(x, y)$. Suppose, for *reductio*, that there exist $a, b \in |G|$ and $\tau \in T(\Delta)$, such that

$$G \models R^G_\tau(a) \wedge R^G_\tau(b) \wedge \neg \phi^{\approx}_{\tau}(a, b).$$

We consider the following cases of τ.

If $\tau = b$, or for some $C \in \mathcal{C}$, $\tau = C$, then by the definition of G and Definition 4.1, it is easy to see that

$$a \approx^G_\tau b \quad \text{iff } a = b.$$

That is, $G \models \phi^{\approx}_{\tau}(a, b)$. This contradicts the assumption.

If $\tau = \{\tau'\}$, then by (†) and $G \models \neg \phi^{\approx}_{\tau}(a, b)$, we have $G \not\models a \approx^G_\tau b$ and

$$G \models \forall x \exists y\, (*(a, x) \rightarrow *(b, y) \wedge x \approx^G_{\tau'} y) \wedge \forall x \exists y\, (*(b, x) \rightarrow *(a, y) \wedge x \approx^G_{\tau'} y).$$

By (†), we have $G \models a \neq b$ and

$$G \models \forall x\, y\, (x \approx^G_{\tau'} y \rightarrow x = y).$$

Therefore,

$$G \models \forall x\, (*(a, x) \leftrightarrow *(b, x)).$$

That is, a and b are identifiable. This contradicts the definition of G.

Similarly, if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then by (†) and $G \models \neg \phi^{\approx}_{\tau}(a, b)$, we have $G \not\models a \approx^G_\tau b$ and for every $i \in [1, n]$,

$$\forall x \forall y\, (l_i(a, x) \wedge l_i(b, y) \rightarrow x \approx^G_{\tau_i} y).$$

By (†), we have $G \models a \neq b$ and for every $i \in [1, m]$,

$$G \models \forall x\, y\, (x \approx^G_{\tau_i} y \rightarrow x = y).$$

Therefore, by $G \models \Phi(\Delta)$, for every $i \in [1, m]$, we have

$$G \models \forall x\, (l_i(a, x) \leftrightarrow l_i(b, x)).$$

That is, a and b are identifiable. This again contradicts the definition of G.

Therefore,

$$G \models \bigwedge_{\tau \in T(\Delta)} \forall x\, y\, \phi^{\approx}_{\tau}(x, y) \wedge \forall x y\, (\bigvee_{\tau \in T(\Delta)} x \approx_\tau y \rightarrow x = y).$$

That is, $G \models \Phi^{\approx}(\Delta)$.

This completes the proof of Proposition 4.9. ∎

4.3 The decidability of word constraint implication in $\mathcal{M}^{\approx}$

Finally, we show Theorem 4.1. More specifically, let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. We show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a small model G of size at most 2^{mN} and $G \in \mathcal{U}_f(\Delta)$, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ.

Consider $\tau = type(lt(\varphi))$. By Lemma 4.5, τ can be one of the following:

1. $\tau \in BC(\Delta)$,
2. $\tau = DBtype$,
3. $\tau = \{\tau'\}$.

For the first case, the filtration argument and identifying operation given above are sufficient to establish the small model property.

Corollary 4.10: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $type(lt(\varphi)) \in BC(\Delta)$ and $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model G in $\mathcal{U}_f(\Delta)$ such that the size of G is at most 2^{mN}, where m and N are as described in Proposition 4.4. ∎

Proof: By Proposition 4.4, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a structure H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

and the the size of H is at most 2^{mN}.

Let G be the ultimately identified structure constructed from H. By Proposition 4.9, the size of G is at most 2^{mN}. By the definition of H given in the proof of Proposition 4.4, we have that for every $\tau \in T(\Delta)$, $H \models \forall x\, y\, (x \approx_\tau^H y \rightarrow x = y)$. Hence again by Proposition 4.9, $G \models \Phi^{\approx}(\Delta)$. In addition, by $H \models \Phi(\Delta)$, we also have $G \models \Phi(\Delta)$.

Since $H \models \Sigma$, by Proposition 4.9, $G \models \Sigma$. In addition, by $H \models \neg\varphi$, there is $a \in |H|$ such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

By Lemma 3.1, $a \in R_\tau^H$ where $\tau = type(lt(\varphi))$. Since $\tau \in BC(\Delta)$, by Definition 4.1, it can be shown by a straightforward induction on i that a is in $|G_i|$ for every $i \in [1, m]$, where G_i is the structure constructed at stage i in the definition of G, as described in Definition 4.3. Indeed, a cannot be identified with any other node in any $|G_i|$. Hence for every $i \in [1, m]$, G_i respects $\neg\varphi$. Therefore, by Proposition 4.9, $G \models \neg\varphi$. ∎

Now assume that $\tau = DBtype$. Let H be the structure constructed in the proof of Proposition 4.4, and G be the ultimately identified structure constructed from H. By Corollary 4.6 and the proof of Proposition 4.4, it is easy to see that $R_{DBtype}^H = \{r^H\}$. Therefore, r^H cannot

be identified with any other node in the construction of G. Thus by Proposition 4.9, G is indeed the small model of $\bigwedge \Sigma \wedge \neg\varphi$ described above.

Finally, let us consider $\tau = \{\tau'\}$. Let H and G be as described above, and G_i be the structure constructed at stage i in the definition of G. Consider node a in $|G_i|$ such that

$$G_i \models lt(\varphi)(r^{G_i}, a) \wedge \neg rt(\varphi)(r^{G_i}, a),$$

It is possible that when constructing G_{i+1}, a is identified with some $b \in |G_i|$ such that $H \models rt(\varphi)(r^{G_i}, b)$. If this happens, then it is possible that $G \models \varphi$.

To prevent this, we need to use a slightly stronger filtration argument. We first give the following definition and lemma, which illustrate the motivation for strengthening the argument given in the proof of Proposition 4.4.

Definition 4.4: Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$, φ be a word constraint in $P_w(\Delta)$ such that $type(lt(\varphi)) = \{\tau'\}$, and H be a $\sigma(\Delta)$-structure. Then H is said to satisfy the *isolation condition with respect to* φ if there exists $a \in |H|$ such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a),$$

and moreover, for any $b \in |H|$ such that $H \models rt(\varphi)(r^H, b)$, there exists $c \in |H|$ such that either

$$H \models *(a, c) \wedge \neg *(b, c),$$

or

$$H \models \neg *(a, c) \wedge *(b, c).$$

■

Lemma 4.11: Let Δ, H, G, Σ and φ be as described in Proposition 4.9. Assume that $H \models \bigwedge \Sigma \wedge \neg\varphi$ and $type(lt(\varphi)) = \{\tau'\}$. Then $G \models \bigwedge \Sigma \wedge \neg\varphi$ if H satisfies the isolation condition with respect to φ. ■

Proof: Since H satisfies the isolation condition with respect to φ, there exists $a \in |H|$, such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a),$$

and moreover, by Definition 4.1, for any $b \in |H|$ such that $H \models rt(\varphi)(r^H, b)$, a and b are not identifiable in H. Hence H respects $\neg\varphi$ when identifying nodes in H. In addition, the structure G_2 resulting from identifying any two identifiable nodes o_1 and o_2 in H also satisfies the isolation condition with respect to φ. To see this, first notice that by Definition 2.1, $lt(\varphi)$ must be of the form $\alpha \cdot K$, where K is a record label and either $type(\alpha) = DBtype$, or $type(\alpha) \in \mathcal{C}$. In both cases, for any node $o \in |H|$ such that $H \models \alpha(r^H, o)$, $o \notin \{o_1, o_2\}$. The same statement also holds for any $o \in |H|$ such that $H \models \alpha \cdot K \cdot *(r^H, o)$. Because of this, by Definitions 4.1 and 4.4, it is easy to verify that identifying o_1 and o_2 does not violate the isolation condition with respect to φ.

In fact, by a straightforward induction on i, it can be shown that every G_i constructed in the definition of G satisfies the isolation condition with respect to φ. Therefore, G_i respects $\neg\varphi$ when identifying identifiable nodes. Thus by Proposition 4.9, $G \models \bigwedge \Sigma \wedge \neg\varphi$. ■

The purpose of introducing a stronger filtration argument is to ensure the isolation condition. We give the argument in two steps. We first convert an arbitrary model of $\bigwedge \Sigma \wedge \neg\varphi$ to a model with certain property. We then refine the notion of *mlb* defined in the proof Proposition 4.4 based on this property.

Recall the notion of *lb* introduced in Section 4.1. For each $G \in \mathcal{U}(\Delta)$, we define an equivalence relation $\sim$ on $|G|$ as follows:

$$a \sim b \quad \text{iff} \quad lb(a,\, G,\, \Sigma \cup \{\varphi\}) = lb(b,\, G,\, \Sigma \cup \{\varphi\}).$$

Let $[o]$ denote the equivalence class of o with respect to $\sim$, and let $[G] = \{[o] \mid o \in |G|\}$.

Definition 4.5: Let Δ and φ be as described in Definition 4.4, G be a $\sigma(\Delta)$-structure, and $a \in |G|$ such that

$$G \models lt(\varphi)(r^G,\, a) \wedge \neg rt(\varphi)(r^G,\, a).$$

Then G is said to have the *semi-isolation property with respect to* φ if it satisfies the following condition: for every $[b] \in [G]$ such that $rt(\varphi) \in lb(b,\, G,\, \Sigma \cup \{\varphi\})$, there exist $c, c' \in |G|$, such that one of the following holds:

- $G \models \neg * (a,\, c)$ and for any $o \in [b]$, $G \models *(o,\, c)$;
- $G \models *(a,\, c)$ and for any $o \in [b]$, $G \models \neg * (o,\, c)$;
- $G \models *(a,\, c) \wedge *(a,\, c')$, and there is a unique $b' \in [b]$ such that $G \models *(b',\, c) \wedge \neg * (b',\, c')$. In addition, for any $o \in [b]$, if $o \neq b'$, then $G \models \neg * (o,\, c)$.

The nodes c and c' are called the *isolating nodes for* a *and* $[b]$. ■

Lemma 4.12: Let $\Delta = (\mathcal{C},\, \nu,\, DBtype)$ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$ such that $type(lt(\varphi)) = \{\tau'\}$. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that $H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$ and H has the semi-isolation property with respect to φ. ■

Proof: Let G be a model of $\bigwedge \Sigma \wedge \neg\varphi$ in $\mathcal{U}(\Delta)$. Then there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G,\, a) \wedge \neg rt(\varphi)(r^G,\, a).$$

By $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$, for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$, there exists $c \in |G|$ such that either $G \models *(a,\, c) \wedge \neg * (b,\, c)$, or $G \models \neg * (a,\, c) \wedge *(b,\, c)$. For each $[b] \in [G]$ such that $rt(\varphi) \in lb(b,\, G,\, \Sigma \cup \{\varphi\})$, we modify G as follows.

- If there exists $c \in |G|$ and $b' \in [b]$ such that $G \models \neg * (a,\, c) \wedge *(b',\, c)$, then for any $o \in [b]$, we add an edge labeled with $*$ from o to c.

- If there exists $c \in |G|$ such that $G \models *(a,\, c)$ and for any $o \in [b]$, $G \models \neg * (o,\, c)$, then c is the isolating node for a and $[b]$.
- Otherwise there must be $c, c' \in |G|$ and $b' \in [b]$, such that $G \models *(a,\, c) \wedge *(a,\, c')$ and $G \models *(b',\, c) \wedge \neg * (b',\, c')$. In this case, for any $o \in [b]$, if $o \neq b'$, then we remove all the edges labeled with $*$ from o to c.

Let H be the structure resulting from modifying G as above. Then it is easy to see that H has the semi-isolation property with respect to φ and $H \models \Phi(\Delta)$. In addition, by Lemma 4.3, it is easy to verify that $H \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Using Lemma 4.12, we present a stronger filtration argument as follows.

Proposition 4.13: Let Δ, Σ and φ be as described in Lemma 4.12. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

H satisfies the isolation condition with respect to φ, and the size of H is at most $2^{2^{mN}}$, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ∎

Proof: By Lemma 4.12, there exists a model G such that $G \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$, and G has the semi-isolation property with respect to φ. Hence there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G,\, a) \wedge \neg rt(\varphi)(r^G,\, a),$$

and for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$, there exist isolating nodes $c_{[b]}, c'_{[b]}$ for a and $[b]$. Recall the notions of mlb and f given in the proof of Proposition 4.4. We refine the definition of mlb such that

- for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$,

  $$mlb(b) = (lb(b,\, G,\, \Sigma \cup \{\varphi\}),\ tag(b,\, c_{[b]}),\ tag(b,\, c'_{[b]})),$$

  where

  $$tag(b,\, o) = \begin{cases} \text{true} & \text{if } G \models *(b,\, o) \\ \text{false} & \text{otherwise} \end{cases}$$

- $mlb(a) = a$,
- for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$, $mlb(c_{[b]}) = c_{[b]}$ and $mlb(c'_{[b]}) = c'_{[b]}$.

Define function f and structure H as in the proof of Proposition 4.4. Similarly, it can be shown that $H \models \Phi(\Delta)$. Moreover, since $m \geq 2$, it is easy to verify that the size of H is at most $2^{2^{mN}}$. In addition, for every $o \in |G|$, it can be verified that

$$lb(o,\, G,\, \Sigma \cup \{\varphi\}) \ = \ lb(f(o),\, H,\, \Sigma \cup \{\varphi\}). \qquad (\ddagger)$$

Hence by Lemma 4.3 and the assumption that $G \models \bigwedge \Sigma \wedge \neg\varphi$, we have $H \models \bigwedge \Sigma \wedge \neg\varphi$.

Next, we show that H satisfies the isolation condition with respect to φ.

By the definition of mlb, for any $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$, $f(c_{[b]}) = c_{[b]}$ and $f(c'_{[b]}) = c'_{[b]}$. We also have $f(a) = a$. In addition, by the the definition of H, it is easy to show the following:

$$\begin{aligned} H \models *(f(b), c_{[b]}) \quad &\text{iff} \quad G \models *(b, c_{[b]}) \\ H \models *(f(b), c'_{[b]}) \quad &\text{iff} \quad G \models *(b, c'_{[b]}) \\ H \models *(a, c_{[b]}) \quad &\text{iff} \quad G \models *(a, c_{[b]}) \\ H \models *(a, c'_{[b]}) \quad &\text{iff} \quad G \models *(a, c'_{[b]}) \end{aligned}$$

Hence $c_{[b]}$ and $c'_{[b]}$ are isolating nodes for a and $[f(b)]$ in H. By ($\ddagger$), we have

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

Moreover, for any $o \in |H|$, if $H \models rt(\varphi)(r^H, o)$, then $o = f(b)$ for some $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$. Therefore, by Definitions 4.4 and 4.5, it can be shown that H satisfies the isolation condition with respect to φ. ∎

The following corollary completes the proof of Theorem 4.1.

Corollary 4.14: Let Δ, Σ and φ be as described in Lemma 4.12. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model H in $\mathcal{U}_f(\Delta)$ such that the size of H is at most 2^{mN}, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ∎

Proof: By Lemma 4.12, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that $H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$ and H has the semi-isolation property with respect to φ. Thus by Proposition 4.13, there is H' such that $H' \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$, H' satisfies the isolation condition with respect to φ, and the size of H' is at most 2^{mN}. Let G be the ultimately identified structure constructed from H'. Then by Lemma 4.11,

$$G \models \bigwedge \Sigma \wedge \neg\varphi.$$

By Proposition 4.9, the size of G is at most 2^{mN}, and in addition,

$$G \models \Phi(D).$$

By the definition of H' given in the proof of Proposition 4.13 (see also the proof of Proposition 4.4), we have that for every $\tau \in T(\Delta)$, $H' \models \forall\, x\, y\, (x \approx_\tau^{H'} y \rightarrow x = y)$. Thus again by Proposition 4.9, we have

$$G \models \Phi^{\approx}(D).$$

Hence $G \in \mathcal{U}_f(\Delta)$. ∎

5 Conclusions

We have provided an answer to the following open question: In the context of an object-oriented data model $\mathcal{M}^{\approx}$ which supports complex value equality, whether is word constraint implication decidable? We have introduced equality constraints, in addition to type constraints, to characterize database schemas in $\mathcal{M}^{\approx}$, and established the decidability of the implication and finite implication problems for word constraints in the presence of the type and equality constraints. Following up [11], which addresses the interaction between type and path constraints, this paper has elaborated the interaction among constraints of three different forms: type, equality and word constraints.

References

[1] S. Abiteboul. "Querying semi-structured data". In *Proceedings of the 6th International Conference on Database Theory*, 1997.

[2] S. Abiteboul and J. Van Den Bussche. "Deep equality revisited". In *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases (DOOD)*, Vol. 1013 of *Lecture Notes in Computer Science,* pp. 213-228, Springer, 1995.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesly Publishing Company, 1995.

[4] S. Abiteboul and P. C. Kanellakis. "Object identity as a query primitive". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 159-173, 1989.

[5] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner. "The lorel query language for semistructured data". *Journal of Digital Libraries*, 1(1), 1997.

[6] S. Abiteboul and V. Vianu. "Regular path queries with constraints". In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[7] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problems*. Springer, 1997.

[8] P. Buneman. "Semistructured data". Tutorial in *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[9] P. Buneman, W. Fan, and S. Weinstein. "Path constraints in the presence of types". Technical Report MS-CIS-97-16, Department of Computer and Information Science, University of Pennsylvania, 1997. Available from `http://www.cis.upenn.edu/~db/langs/97papers.html`.

[10] P. Buneman, W. Fan, and S. Weinstein. "Path constraints on semistructured and structured data". In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 1998. Available also in `http://www.cis.upenn.edu/~db/langs/98papers.html`.

[11] P. Buneman, W. Fan, and S. Weinstein. "Interaction between path and type constraints". Technical Report MS-CIS-98-16, Department of Computer and Information Science, University of Pennsylvania, 1998. Available from `http://www.cis.upenn.edu/~db/langs/98papers.html`.

[12] R. G. G. Cattell (ed.). *The object-oriented standard: ODMG-93* (Release 1.2). Morgan Kaufmann, San Mateo, California, 1996.

[13] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[14] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesly Publishing Company, 1979.

[15] Anthony Kosky. *Transforming databases with recursive data structures*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.